REMARKS

Claims 1-13 are currently pending in the application. The foregoing separate sheets marked as "Listing of Claims" shows all the claims in the application, with an indication of the current status of each.

The Examiner's withdrawal of the rejection under 35 U.S.C. §112, second paragraph, is acknowledged with appreciation.

The Examiner maintains a requirement that the applicant submit a replacement oath or declaration. However, it appears that the original Oath and Declaration provided the inventor's signature over his address, which was his residence and post office address, as required. It was not expressly stated in the Declaration, below the residence address, that the post office address was the same as the residence address. Following a telephone call to the Examiner on April 27, 2005, which clarified the foregoing facts, the undersigned respectfully requests that the original Oath and Declaration be accepted as compliant with the requirement that the inventor's signature appear over his residence and post office address.

The Examiner maintains rejection of claims 1-13 under 35 U.S.C. §102(a) as being anticipated by U.S. Patent No. 6,591,272 to Williams. The Examiner contends that Williams discloses each and every limitation required by the claims, and asserts that the foregoing argument fails to describe how the recited portions of Williams do not read on the required claim limitations. It is respectfully submitted that Williams fails to disclose "each and every limitation required by the claims" and that the recited portions of Williams do not "read on the required claim limitations." This will now be shown with reference to the following table of claim 1, showing each element of the claim, as stated in the claim language and also showing the reference cited by the Examiner in rejecting the claim under §102.

Claim Element	Claim language	Examiner's Reference to Williams
Claim 1: preamble	An automated method for converting tables of data in a source database to components of a **target application**	a) col. 4, lines 48-50
obtaining	identification of a) specified tables in said source database containing data usable in said **target application**, b) a target location for said **target application** and c) an application server being used for development of said **target application** at said target location	b) col. 44, lines 52-53; Fig. 16 and associated text c) Fig. 17 (option for output directory) d) col. 30, line 56
reading	definitions of said specified tables from said source database; and	e) col. 4, lines 52-66
generating	from said database definitions a plurality of source code files in a language of said **target application**, each said specified table being referenced consistently across said plurality of source code files, said plurality of source code files including object classes and deployment descriptors, said specified tables being made accessible to a remote client by said **target application**, said **target application** being developed using said plurality of source code files.	f) col. 4, lines 52-66 col. 4, lines 52-66 g) Fig. 3, items 12 and 16 Fig. 3, items 12 and 16

As the above table highlights, the claims are drawn to a **target application** whose preparation is directly supported by the invention. Williams fails to address a target application, instead addressing **objects**. For example, in the Examiner's reference cited in item a) in the above table, Williams talks about "familiar and customary **objects**" and not about a **target application**. Similarly, in item b) Williams talks about "distributed **objects**", and the text surrounding Fig. 16 discusses the generation of distributed **objects** from the selected tables. In item c), Williams describes choice of an output directory, but this is for the **objects**, in their intermediate state. The OSF system established by Williams has as its purpose a cross platform formulation of objects, in a "normalized" view that is usable by an development platform. But this requires – as part of the Williams invention – a translation convention for each such platform. The present invention is not concerned with such a translation scheme, and proceeds directly to the **target application**. In item d), the reference mentions the term "application server" but as the locus for housing "the OSF-built Enterprise Java Beans. No mention is made of development of a **target application** on the application server. Item e) is a valid reference for the reading of tables from the source database, which is done in both Williams and the present invention.

However, the reference in item f) refers not to "a language of said **target application**" as in the present invention but rather to "a vendor neutral standardized view of the database schemas." Clearly, a §102 rejection cannot be sustained because Williams explicitly chooses to develop intermediate objects in a normalized format, not in the claimed "language of said target application." Again, this is consistent with the purpose of Williams, which is to provide an intermediate standardized format that can then be translated for use on any particular development platform. In effect, Williams admits that these intermediate objects are not in the "language of said target application.

In the reference of item g) the architecture described by Williams provides for access by a remote client, but there is no discussion in Williams of a **target application**. Here, too, this is understandable because Williams provides an intermediate standardized format. By contrast, the present invention provides for a direct generation for a **target application**.

Williams does not disclose the simple and direct connection between the present invention and a **target application**. When there are changes in the source database, the invention is simply run again, automatically updating name references across the plurality of source code files being used in the target application. Williams provides for updating of its objects, but these are in an intermediate standardized format. That updating step does not apply to the **target application** directly, as in the present invention, but in Williams requires a further translation step to convert the intermediate **object** for use by a particular language.

Consequently, the Examiner is incorrect in asserting that the Williams disclosure directed to generic **objects** in an intermediate standardized format reads upon the **target application** described in the present invention. In particular, it is clear that these generic **objects** are not in "the language of said **target application**." This failure of the Examiner's §102 rejection carries through to remaining claims 2-13, which depend from claim 1.

Williams appears to be a method for a) reading the definitional elements of a database to determine data types and interrelationships between relational data elements, b) assembling them into a vendor neutral standardized view of the database schemas, and c) creating all the possible logical objects that are in the database. A user would then select a subset of these objects for an application. The problem being addressed by Williams is "the inherent mismatch between data stored in relational databases and the format and structure of this relational data in object based systems" (col. 3, lines 25-28). Usually, it takes an expert of considerable skill to manually map database tuples into objects, even using object-relational mapping tools. Under the

solution provided by Williams, this mapping can now be done by an inexperienced user.

Before considering the differences between Williams and the present invention, it is helpful to consider what the two inventions have in common: both are methods of converting databases to a standardized format. In Williams, the conversion is from a specific query language (SQL) database to a standardized format, which serves to facilitate ultimate conversion at the client side to a desired format. The standardized format is not itself the end product of the conversion desired by the user. The purpose of the standardized format is to enable inexperienced users to obtain appropriate objects for use in an application at a client. Creation of these usable objects is done at runtime, using source code (which has been merged into the standardized view) to encapsulate the metadata and data values of the object desired by the user. The client side user is then able to assemble the object into the format required by his application. It should be noted that the objects are source code segments composed of functionality and data. It should further be noted that Williams adheres to the Common Object Request Broker Architecture (CORBA), which is supported by a consortium known as the Object Management Group and whose Interface Definition Language (IDL) is used by clients to invoke an operation on an object on a remote server. IDL is independent of programmng language, but maps to all popular programming languages (e.g. C, C++, Java, COBOL, Smalltalk, Ada, Lisp, etc.).

By contrast, the present invention does not use an intermediate standardized format. Although a Standard Query Language (SQL) example is described in detail in the specification, the invention is not limited to that example. However, according to the methodology of the invention, an equivalent to the "bean grinder" would be developed for different database examples. In any case, the target for the conversion is the Enterprise Java model, and the described and claimed methodology is usable to convert any database to the Enterprise Java model. In contrast to Williams, the

purpose of the present invention is to avoid the labor intensive exercise which even experienced software engineers must undergo under the prior art to convert a database to the Java model. The methodology for accomplishing this purpose has several important characteristics. First, it reverses the conventional method of conversion by working from the database specification to the target Java model. The conventional approach would be for the application developer – and it is important to emphasize that the "user" of the present invention is the developer of an Enterprise Java application – to first write the application in Java, and then determine how the database can be mapped to the target Enterprise Java model.

It should be noted that Williams is similar in this respect, in that it begins with the database to be converted. However, Williams has no alternative since the target database is not known. Instead, Williams converts a SQL database to an intermediate standardized form. The structure of the standardized form (metadata coupled with skeleton code templates representative of final classes) avoids the requirement that the user have a high level of expertise in order to deal with the "inherent mismatch between data stored in relational databases and the format and structure of this relational data in object based systems" (see above citation).

The methodology of the present invention begins with the recognition that simply "automating" the laborious and complex task presented by the conventional method is not adequate. For even one table in a database, converting that table into usable source code in Enterprise Java's object oriented approach to providing functionality and data to remote users over the Internet requires that "numerous files" of Enterprise Java source code be generated (page 6, line 19). In the conventional approach, the developer then uses the Enterprise Java deployment descriptor file to back his way into the name assignments necessary to coordinate the database table among the "numerous files" in the Enterprise Java model. The problem is that this is an error prone process, and furthermore must be reviewed each time there is a change in the database itself. The insight of the present invention is that it makes more sense

– however counterintuitive – to begin with the database tables and generate from the database the "numerous files" required by the Enterprise Java model at the beginning, before the developer writes code for the Enterprise Java application. This reversal of order is what is referred to in the specification (page 6, lines 11-12) as "reverses the conventional method of conversion by working from the database specification to the EJB".

It turns out that the thorough and methodical use of metadata in the source database, as described in great detail in Figures 3-9 and accompanying text, is both necessary and sufficient to automatically accomplish the coordination of names and functions among the "numerous files" of the Enterprise Java model in order to make a particular database table available for use in the Enterprise Java application being developed by the developers. The apparently rote nature of what is described in the application is misleading. A more selective approach – simply "automating" what is typically undertaken by the conventional method – results in source code whose viability cannot be tested until runtime, that is, until after the developed application is deployed (see page 2, lines 10-12). This is not an acceptable situation, and is the reason why the reverse approach of the present invention was pursued. It should be emphasized again that the approach of the present invention is to proceed directly and without an intermediate structure (as the "standardized format" of Williams) to source code usable at run-time. By generating the plurality of Enterprise Java files necessary for a database table to be usable in an Enterprise Java application, and doing so from the database so that these files can be used as components in the application which is then developed, the necessary coordination between table names and function names among these plurality of files is thereby assured. It is not self-evident why this should be so, but it is.

The test of this proposition is its use during development of the Enterprise Java application. Whenever the source database is changed, the automated method (which is implemented as a "bean grinder") is simply repeated and the "numerous

files" are produced again, with the confidence that the table names and function names will be consistent across these "numerous files." The process is linear and direct, with all necessary information being gathered at the outset, thereby promoting a high-speed automated solution. It is important to note that the method of the invention – implemented for the Enterprise Java model as the "bean grinder" – results in actual EJB source code components, rather than an intermediate structure as in Williams.

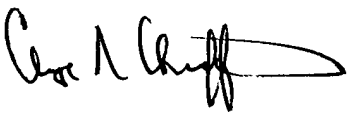
The significance of coordination between table references – table names and the names of functions providing desired functionality – across the plurality of files may be seen from the specification. For example, the "table name" appears across these files (see page 13, line 3, for DATA file; page 14, line 10, for HOME file; page 15, lines 18-19, for Remote file; page 17, lines 1-2, for Bean file; page 20, line 4, for PrimaryKey file; page 21, line 14, for Persistent file; page 27, line 2, for methods listed in the Deployment Descriptor file; and page 27, line 11, for container managed transactions listed in the Deployment Descriptor file for methods; and page 29, line 7, for the file specifying the target application server). Similarly, the functionality embodied in methods also appears in more than one file. For example, the "create" function appears in both the home interface file (page 12, line 13) and the persistent file (page 12, line 15).

In view of the foregoing, it is requested that the application be reconsidered, that claims 1-13 be allowed, and that the application be passed to issue.

Should the Examiner find the application to be other than in condition for allowance, the Examiner is requested to contact the undersigned at 703-787-9400 (fax: 703-787-7557; email: clyde@wcc-ip.com) to discuss any other changes deemed necessary in a telephonic or personal interview.

If an extension of time is required for this response to be considered as being timely filed, a conditional petition is hereby made for such extension of time. Please charge any deficiencies in fees and credit any overpayment of fees to Attorney's Deposit Account No. 50-2041.

Respectfully submitted,



Clyde R Christofferson
Reg. No. 34,138

Whitham, Curtis & Christofferson, P.C.
11491 Sunset Hills Road, Suite 340
Reston, VA 20190
703-787-9400
703-787-7557 (fax)

Customer No. 30743